REMARKS

Applicants respectfully request reconsideration and allowance of the present application. By this Amendment, Applicants amend the specification and claims 8, 10, 12, 14, 19-21, 32, 35, 36, 38, 42-44, 50, and new claims 51-52 have been added. Claims 1-52 will be pending in the application upon entry of this Amendment.

Information Disclosure Statement

As suggested by the Office Action, an Information Disclosure Statement including non-patent literature listed in the patent specification is being submitted.

Priority

As suggested by the Office Action, page 1 of the specification has been amended to include specific references to earlier-filed applications to which Applicants request priority. Applicants further note that information regarding the earlier-filed applications was also provided on the transmittal sheet accompanying the originally-filed present application in accordance with the Rules.

Claim Objections

Claims 19-21 stand objected to for informalities. The phrase "modulo variable expansion (MVE)" has been inserted into the claims herewith as suggested by the Office Action.

Claim 38 stands objected to under 37 CFR 1.75(c) as being in improper form. The claim dependency has been corrected herewith as suggested by the Office Action.

Having made the corrections suggested by the Office Action the objections to the claims should be withdrawn.

Additional Claim Informalities

Claims 8, 10, 12, 14, 36 and 44 have also been amended for consistency with the term "initiation interval register" in the specification as originally filed. Claims 42 and 50 have been amended to cure informalities noted in the claims as originally filed.

Rejection of Claims Under 35 U.S.C. § 102(e)

In the Office Action, claims 1-3 under 35 USC 102(e) stand rejected as allegedly being taught by U.S. Patent No. 6,085,315 to Fleck et al. ("Fleck"). For reasons set forth more fully below, this rejection is respectfully traversed.

Independent Claim 1 Patentably Defines Over Fleck

Independent claim 1 requires, *inter alia*, "a buffer in the dispatch stage coupled to the functional unit adapted to store a plurality of the instructions before issue to the functional unit." The Office Action indicates that this claimed subject matter is found Fleck. Applicants respectfully disagree.

First, Fleck's loop cache 3 stores instructions to be fed to the "instruction providing unit." If anything in Fleck could be considered, *arguendo*, as corresponding to the claimed invention, the "instruction providing unit," which is represented by instruction demux 7, corresponds to the dispatch stage of the processor. In particular, in Figure 1 of Fleck, the loop cache feeds into the Instruction Demux (Box 7) via the Mux (Box 5). Most importantly, Fleck's loop cache is in a stage **before** the instruction demux. Fleck at col. 3, lines 1-8 explicitly refers to this unit as the "fetch stage." Accordingly, Fleck's loop cache is not **in the dispatch stage** of the processor. Rather, Fleck's loop cache is **before** the dispatch stage (i.e. instruction demux 7), and feeds instructions **to the dispatch stage**, not to the functional units.

This distinction is made even more clear by the plain language of claim 1. Claim 1 requires that the instructions stored in the buffer must be capable of being executed by the functional unit. This is required by antecedence wherein the functional unit is defined as being "adapted to execute **an instruction**," and the claimed buffer must store "a plurality of **the instructions**." Therefore the "instruction" of the claim is an instruction in a form that is capable of being executed by the functional unit.

In contrast, Fleck states in col. 1 lines 50-54 that the loop cache stores instructions to be fed to the "instruction providing unit." This means that the loop buffer of Fleck holds raw instructions that are of the same form as the ones fetched out of the Cache Subsystem (Box 13). The dispatch stage buffer of the invention of claim 1, in contrast, feeds instructions to the functional unit, an example of which is illustrated as box 512 in Figure 5 of the present specification. As shown in Figure 5, multiple buffers are provided, one per functional unit. Claim 1 requires that instructions in the dispatch stage buffers have already

been processed and are stored already "muxed" (in Fleck's terms) or routed to their appropriate functional unit. In contrast, Fleck's alleged "buffer" (i.e. loop cache) does not store instructions in a form that are capable of being issued to the functional unit. Rather, Fleck further requires "instruction demux 7" to put the instructions in a form that are capable of being executed by alleged functional units 10, 11 and 12. Therefore, Fleck's "loop cache" does not meet the limitations of the dispatch stage buffer of claim 1, and claim 1 patentably defines over Fleck.

For at least the foregoing reasons, the § 102 rejection of claim 1, along with claims 2 and 3 that depend therefrom, should be withdrawn.

Claim 3 Further Patentably Defines Over Fleck

Claim 3 depends from claim 1, which has been shown above to patentably define over Fleck. Accordingly, claim 3 is patentable for at least the reasons claim 1 is patentable.

Claim 3 further requires "control logic coupled to the buffer adapted to cause a certain one of the stored plurality of instructions to be issued to the functional unit in accordance with a loop iteration stage." The Office Action indicates that this subject matter is met by Fleck (column 2, lines 45-67; column 4, lines 9-56; Figure 1; and Figure 3). Applicants respectfully disagree.

First, similarly as set forth above, whereas the claim requires that the control logic identifies one of the **stored** instructions to be issued to **the** identified functional unit, Fleck's alleged buffer (i.e. "loop cache 3") is positioned before the instruction demux 7. Therefore, the structure is not associated with any particular function unit, and instructions stored in loop cache 3 cannot be issued without change to any functional unit.

Second, Fleck's loop cache is accessed by Program Counter 2 (instruction address). However, claim 3 requires that the instructions are issued to the functional unit from the buffer according to a loop iteration stage. This would require "PC update and control 2" to correspond to the claimed loop iteration stage, a concept which is not taught or suggested by Fleck.

For at least the foregoing reasons, the § 102 rejection of claim 3 should be withdrawn.

Rejection of Claims Under 35 U.S.C. § 103(a) By Fleck and Subramanian

Claims 4-18, 26-33, 35-41 and 43-49 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over Fleck in view of U.S. Patent No. 5,867,711 to Subramanian et al. ("Subramanian"). For reasons set forth more fully below, this rejection is respectfully traversed.

Claims 4-18 Patentably Define Over Fleck and Subramanian By Dependence On Claim 1

Claims 4-18 depend from claim 1, and certain other claims further depend directly or indirectly from claim 3, both of which have been shown above to patentably define over Fleck. The alleged combination with Subramanian would not have suggested the subject matter of claims 1 and 3 that is not missing from Fleck, and the Office Action correctly fails to allege as such. Accordingly, claims 4-18 are patentable for at least the reasons claims 1 and 3 are patentable.

Claim 4 Further Patentably Defines Over Fleck and Subramanian

Claim 4 depends from patentable claims 1 and 3 and further requires that the control logic causes the stored instructions to be issued from the buffer to the functional unit "in accordance with **a cycle within the loop iteration stage**." The invention of claim 4 thus requires a measure of the number of cycles (i.e. "loop cycles") in the loop body, whereas neither Fleck nor Subramanian teaches or suggests such a measure to be tracked by the control logic in a processor.

First, it is important to note that Subramanian teaches compiler software methods for creating a modulo schedule of a loop iteration. Thus, Subramanian does not teach, and in fact **teaches away** from, providing any control logic **in a processor** adapted to cause the certain ones of the instructions to be issued **in accordance with a cycle within the loop iteration stage** (see the present specification at, for example, page 3, line 19 to page 4 line 3, page 6, line 22 to page 7 line 4, and page 7, lines 14-20).

Subramanian aims to derive the loop stages and cycle schedule from a dependence graph. As taught by Subramanian, the schedule is then represented in an instruction sequence that consists of a Prolog, Kernel and Epilog (PKE form). In Subramanian's teaching, the prolog and epilog code sequences are explicitly generated by the compiler

software and provided to the processor in that form. The target processor hardware itself is not aware of the concept of loop iteration stages and loop cycles.

In stark contrast, the present invention provides a hardware apparatus that can allow the compiler software to efficiently communicate the number of loop iteration stages and cycles of its derived schedule to hardware. In accordance with the present invention, a compiler is not required to explicitly generate entire instruction sequences PKE form. Rather, the invention allows the compiler to compactly represent the schedule in the instruction stream. The control unit then dynamically derives the loop iteration stage and cycles information needed to cause the appropriate instructions in each buffer to be issued to the associated function unit.

To summarize, Subramanian and its associated prior art teach how to reach a schedule from a dependence graph in compiler software, whereas the present invention teaches how to enable compact representation of the derived schedule and efficient hardware buffering and execution.

As set forth above, claim 4 requires control logic in the processor that maintains a loop cycles counter as a pointer that iterates through the buffer. The claimed control logic uses this pointer to issue successive instructions out of the buffers. In contrast, Fleck relies on the program counter values. Accordingly, even if one could apply the teachings of Subramanian to Fleck, the alleged combination would still not have suggested the subject matter of claim 4 because the alleged combination would still have only allowed entire loop instruction sequences (downloaded from Subramanian's compiler solution and stored in Fleck's cache subsystem) to be issued by Fleck's processor in accordance with program counter values. The invention of claim 4, however, requires instructions to be issued using control logic in the processor which causes them to be issued in accordance with a cycle within the loop iteration stage.

For at least the foregoing reasons, claim 4, together with claim 7 that depends therefrom, patentably define over Fleck and Subramanian and the § 103 rejection thereof should be withdrawn.

Claims 15 and 16 Further Patentably Defines Over Fleck and Subramanian

Claims 15 and 16 ultimately depend from patentable claim 1 and further require that the control logic is operative "so that the fetch unit can be shut down during execution of the

stored plurality of instructions." In contrast, Fleck uses Program Counter 2 to access Loop Cache 3, **which are part of the fetch unit 1**. As a result, the fetch unit cannot **possibly** be shut down during the execution of the stored plurality of instructions in Loop Cache 3. In the inventions recited in claims 15 and 16, in contrast, because the buffers are located in the dispatch stage (i.e. on the opposite side of Fleck's Instruction Demux) and accessed by a cycle index generated by the control logic, and not the Program Counter, the instruction fetch unit (including, for example, logic corresponding to Fleck's PC Update and Control 2 as well as the Instruction Demux) can be completely shut off during the execution of the stored plurality of instructions.

For at least the foregoing reasons, claims 15 and 16 patentably define over Fleck and Subramanian and the § 103 rejection thereof should be withdrawn.

Independent Claim 26 Patentably Defines Over Fleck and Subramanian

Independent claim 26 requires that:

- The buffer is "in the dispatch stage";
- The buffer is "associated with a functional unit";
- The buffer has "a first portion for storing a kernel set of instructions"; and
- The buffer has "a second portion for storing a plurality of modulo schedule stage identifiers"

As set forth in more detail above in connection with claim 1, Fleck's alleged buffer (loop cache 3) is not "in the dispatch stage," but before it (i.e. instruction demux 7).

Moreover, Fleck's loop cache 3 is not "associated with a functional unit," but rather serves all of the functional units 10, 11 and 12.

In still further contrast, Fleck's loop cache 3 does not contain the first and second portions required by claim 26. The Office Action admits this, but relies on Subramanian for this subject matter.

First, as set forth more fully above in connection with claim 4, Subramanian teaches compiler software methods for creating a modulo schedule of a loop iteration, and thus **teaches away** from, providing any modulo scheduling logic **in a processor**. Rather, in Subramanian's teaching, the prolog, kernel and epilog code sequences are explicitly generated by the compiler software and provided to the processor in that form. The target

processor hardware itself is not aware of the concept of loop iteration stages and loop cycles.

In stark contrast, the present invention provides a hardware apparatus that can allow the compiler software to efficiently communicate the number of loop iteration stages and cycles of its derived schedule to hardware. In accordance with the present invention, a compiler is not required to explicitly generate entire instruction sequences PKE form. Rather, the invention allows the compiler to compactly represent the schedule in the instruction stream.

As set forth above, claim 26 requires a buffer in the processor having separate portions for storing:

- A kernel set of loop instructions; and
- A plurality of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions.

Subramanian teaches away from providing such a buffer in the processor, and in any event does not teach a buffer having the required first and second portions. Accordingly, even if one could apply the teachings of Subramanian to Fleck, the alleged combination would still not have suggested the subject matter of claim 26.

For at least the foregoing reasons, claim 26, together with claims 27-31 that depend therefrom, patentably define over Fleck and Subramanian and the § 103 rejection thereof should be withdrawn.

Amended Independent Claim 32 Patentably Defines Over Fleck and Subramanian

Amended independent claim 32 requires that:

- The buffers are in "the dispatch stage";
- The buffers are "respectively associated with" the plurality of functional units;
- The buffers in the processor "store the kernel set of loop instructions"; and
- Control logic in the processor is coupled to the buffers "for causing the stored kernel set of instructions to be selectively issued to the functional units".

As set forth in more detail above in connection with claim 1, Fleck's alleged buffer (loop cache 3) is not "in the dispatch stage," but before it (i.e. instruction demux 7).

Moreover, Fleck's loop cache 3 is not a plurality of buffers "respectively associated with" the plurality of functional units, but rather this one cache serves all of the functional units 10, 11 and 12.

In still further contrast, as set forth more fully above in connection with claim 4, Subramanian teaches **compiler software** methods for creating a modulo schedule of a loop iteration, and thus **teaches away** from, providing any modulo scheduling control logic **in a processor**. Rather, in Subramanian's teaching, the prolog, kernel and epilog code sequences are explicitly generated by the compiler software and provided to the processor in that form. The target processor hardware itself is not aware of the concept of loop iteration stages and loop cycles.

In stark contrast, the present invention provides a hardware apparatus that can allow the compiler software to efficiently communicate the number of loop iteration stages and cycles of its derived schedule to hardware. In accordance with the present invention, a compiler is not required to explicitly generate entire instruction sequences PKE form. Rather, the invention allows the compiler to compactly represent the schedule in the instruction stream.

As set forth above, claim 32 requires a buffer and associated control logic **in the processor** wherein:

- The buffers store a kernel set of loop instructions; and
- The control logic causes the stored kernel set of instructions to be selectively issued to the functional units based on the stored kernel set of loop instructions.

Subramanian teaches away from providing such buffers and control logic **in the processor**, and in any event does not teach the buffers and control logic required by claim 32. Accordingly, even if one could apply the teachings of Subramanian to Fleck, the alleged combination would still not have suggested the subject matter of claim 32.

For at least the foregoing reasons, claim 32, together with claims 33 and 34 that depend therefrom, patentably define over Fleck and Subramanian and the § 103 rejection thereof should be withdrawn.

Amended Independent Claims 35 and 43 Patentably Define Over Fleck and Subramanian

Amended independent claims 35 and 43 require that:

- The kernel set of loop instructions are stored in "a dispatch stage" of the processor;
- Loop parameters are stored "in control logic of the processor"; and
- The stored kernel set of loop instructions are caused "to be selectively issued to functional units of the processor" in accordance with the stored loop parameters.

As set forth in more detail above in connection with claim 1, Fleck's alleged buffer (loop cache 3) is not "in the dispatch stage," but before it (i.e. instruction demux 7).

In still further contrast, as set forth more fully above in connection with claim 4, Subramanian teaches **compiler software** methods for creating a modulo schedule of a loop iteration, and thus **teaches away** from, providing any modulo scheduling control logic **in a processor**. Rather, in Subramanian's teaching, the prolog, kernel and epilog code sequences are explicitly generated by the compiler software and provided to the processor in that form. The target processor hardware itself is not aware of the concept of loop iteration stages and loop cycles.

In stark contrast, the present invention provides a hardware apparatus that can allow the compiler software to efficiently communicate the number of loop iteration stages and cycles of its derived schedule to hardware. In accordance with the present invention, a compiler is not required to explicitly generate entire instruction sequences PKE form. Rather, the invention allows the compiler to compactly represent the schedule in the instruction stream.

As set forth above, claim 35 and 43 require storing loop instructions and associated control logic **in the processor** wherein:

- The dispatch stage includes means for storing a kernel set of loop instructions; and
- Control logic stores loop parameters for causing the stored kernel set of instructions to be selectively issued to the functional units based on the stored kernel set of loop instructions.

Subramanian teaches away from providing such storage and control logic **in the processor**, and in any event does not teach the storage and control logic required by claims 35 and 43. Accordingly, even if one could apply the teachings of Subramanian to Fleck, the alleged combination would still not have suggested the subject matter of claims 35 and 43.

For at least the foregoing reasons, claims 35 and 43, together with claims 36-42 and 44-50 that respectively depend therefrom, patentably define over Fleck and Subramanian and the § 103 rejection thereof should be withdrawn.

Rejection of Claims Under 35 U.S.C. § 103(a) By Fleck, Subramanian and Valluri

The Examiner has rejected claims 19-21 under 35 USC 103(a) as allegedly being unpatentable over Fleck in view of Subramanian as applied to claims 9, 11 and 13, above, and further in view of Valluri and Govindarajan's "Modulo-Variable Expansion Sensitive Scheduling" published in *High Performance Computing*, 1998.

Claims 19-21 depend from claims 9, 11 and 13, which have been shown above to patentably define over Fleck and Subramanian, at least because Fleck and Subramanian do not suggest a buffer in the dispatch stage as required by independent claim 1. The further alleged combination of Valluri with Fleck and Subramanian would not cure this deficiency.

Moreover, Valluri is directed to compiler-based scheduling of code, which further teaches away from the invention, which is based in hardware. Accordingly, one skilled in the art would not be led to the hardware-based invention of claims 19-21, even if, *arguendo*, Valluri could be combined with Fleck and Subramanian.

For at least these reasons, the rejections of claims 19-21 should be withdrawn.

Rejection of Claims Under 35 U.S.C. § 103(a) By Fleck, Subramanian and Morrison

Additionally, claims 22-25, 34, 42 and 50 were rejected under 35 USC 103(a) as allegedly being unpatentable over Fleck in view of Subramanian as applied to claims 3, 8, 11 and 13, above, and further in view of U.S. Patent No. 6,421,744 to Morrison et al. ("Morrison").

Claims 22-25 depend ultimately from independent claim 1, claim 34 depends from independent claim 32, claim 42 depends from independent claim 35, and claim 50 depends from independent claim 43. These independent claims have been shown above to patentably define over Fleck and Subramanian. The further alleged combination of these references with Morrison would not overcome the shortcomings of Fleck and Subramanian as discussed above. Accordingly, claims 22-25, 34, 42 and 50 are patentable at least for the reasons claims 1, 32, 35 and 50 are patentable.

Moreover, Morrison does not teach the type of loop iteration required in the rejected claims. For example, claim 34 requires the control logic to be "operative to allow interrupts to be handled at the end of a current one of the number of loop iterations, and to complete the number of loop iterations after the interrupt is handled." In contrast Morrison merely discloses taking interrupts at the end of loop iterations. In modulo scheduled code according to the present invention, there is no natural and clean end of an iteration; all the iterations are overlapped. Accordingly, Morrison's interrupts would not meet the limitations explicitly required by the claims.

For the foregoing reasons, claims 22-25, 34, 42 and 50 patentably define over the cited prior art and the § 103 rejection of the claims should be withdrawn.

Conclusion

All objections and rejections having been addressed, and in view of the foregoing, the claims are believed to be in form for allowance, and such action is hereby solicited. If any points remain in issue which the Examiner feels may be best resolved through a personal or telephone interview, he or she is kindly requested to contact the undersigned at the telephone number listed below.

Respectfully submitted
PILLSBURY WINTHROP LLP

By _____
Mark J. Danielson, Reg. No. 40,580
650-233-4777

REPLY TO CUSTOMER NO. 27498